UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
WACO DIVISION

TELEPUTERS, LLC, Plaintiff v. FUJITSU AMERICA, INC., FUJITSU SEMICONDUCTOR AMERICA, INC., FUJITSU COMPUTER PRODUCTS OF AMERICA, INC., FUJITSU LIMITED and FUJITSU FRONTECH NORTH AMERICA, INC., Defendants	**Case No. 6:20-cv-00640** **JURY TRIAL DEMANDED**

ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff Teleputers, LLC ("Plaintiff" or "Teleputers") hereby files this Original Complaint for Patent Infringement against Defendants Fujitsu America, Inc., Fujitsu Semiconductor America, Inc., formerly known as Fujitsu Microelectronics America, Inc., Fujitsu Computer Products of America, Inc., Fujitsu Limited and Fujitsu Frontech North America, Inc. (collectively "Defendant" or "Fujitsu Entities") and alleges, on information and belief, as follows:

THE PARTIES

1. Teleputers, LLC is a limited liability company organized and existing under the laws of the State of New Jersey with its principal place of business in Princeton, New Jersey.

2. On information and belief, Fujitsu America, Inc. is a California corporation, having its principal place of business at 1250 East Arques Avenue, Sunnyvale, California 94085. Fujitsu

America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu America, Inc. does business in the State of Texas and in the Western District of Texas.

3. On information and belief, Fujitsu Microelectronics America, Inc. is a California corporation, having its principal place of business at 1250 East Arques Avenue, Sunnyvale, CA 94085. Upon information and belief, on or around April 19, 2010, Fujitsu Microelectronics America, Inc. changed its name to Fujitsu Semiconductor America, Inc. Fujitsu Semiconductor America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu Semiconductor America, Inc. does business in the State of Texas and in the Western District of Texas.

4. On information and belief, Fujitsu Computer Products of America, Inc. is a California corporation, having its principal place of business at 1255 East Arques Avenue, Sunnyvale, California 94085. Fujitsu Computer Products of America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu Computer Products of America, Inc. does business in the State of Texas and in the Western District of Texas.

5. On information and belief, Fujitsu Frontech North America, Inc., is located at 2801 Network Blvd, Frisco, Texas 75034. Fujitsu Frontech North America, Inc. can be served with process through its registered agent, C T Corporation System, located at 1999 Bryan St., Ste. 900, Dallas, Texas 75201. Fujitsu Frontech North America, Inc. does business in the State of Texas and in the Western District of Texas.

6. On information and belief, Defendant Fujitsu Limited is a company organized under the laws of Japan with its principal place of business at Shiodome City Center, 1-5-2 Higashi-

Shimbashi, Minato-ku, Tokyo 105-7123, Japan. Fujitsu Limited may be served through its U.S. subsidiaries, *supra*.

JURISDICTION AND VENUE

7. This action arises under the patent laws of the United States, 35 U.S.C. § 1, *et seq*. This Court has subject matter jurisdiction under 28 U.S.C. §§ 1331 and 1338(a).

8. Defendants have committed acts of infringement in this judicial district.

9. On information and belief, Defendants maintain regular and systematic business interests in this district and throughout the State of Texas including through their representatives, employees and physical facilities.

10. On information and belief, the Court has personal jurisdiction over Defendants because Defendants have committed, and continue to commit, acts of infringement in the State of Texas, have conducted business in the State of Texas, and/or have engaged in continuous and systematic activities in the State of Texas. On information and belief, Defendants' accused instrumentalities that are alleged herein to infringe were and continue to be used, imported, offered for sale, and/or sold in the Western District of Texas.

11. On information and belief, Defendants voluntarily conduct business and solicit customers in the State of Texas and within this District, including, but not limited to, its offices located at 13915 Burnet Road, #400, Austin, Texas 78728. *See, e.g.:*

On information and belief, Defendants generate substantial revenue within this District and from the acts of infringement as carried out in this District. As such, the exercise of jurisdiction over Defendants would not offend the traditional notions of fair play and substantial justice.

12. Venue is proper in the Western District of Texas pursuant to 28 U.S.C. § 1400(b) and 28 U.S.C. § 1391(c)(3).

NOTICE OF TELEPUTERS' PATENTS

13. Teleputers is owner by assignment of U.S. Patent No. 6,922,472 ("the '472 Patent") entitled "Method and system for performing permutations using permutation instructions based on butterfly networks." A copy may be obtained at: https://patents.google.com/patent/US6922472B2/en.

14. Teleputers is owner by assignment of U.S. Patent No. 6,952,478B2 ("the '478 Patent") entitled "Method and system for performing permutations using permutation instructions based on modified omega and flip stages." A copy may be obtained at: https://patents.google.com/patent/US6952478B2/en.

15. Teleputers is owner by assignment of U.S. Patent No. 7,092,526B2 ("the '526 Patent" and collectively with the '478 Patent, "the Patents-in-Suit") entitled "Method and system for

performing subword permutation instructions for use in two-dimensional multimedia processing." A copy may be obtained at: https://patents.google.com/patent/US7092526B2/en.

16. Teleputers is owner by assignment of U.S. Patent No. 7,174,014B2 ("the '014 Patent") entitled "Method and system for performing permutations with bit permutation instructions." A copy may be obtained at: https://patents.google.com/patent/US7174014B2/en.

17. Teleputers is owner by assignment of U.S. Patent No. 7,519,795B2 ("the '795 Patent") entitled "Method and system for performing permutations with bit permutation instructions." A copy may be obtained at:

https://patents.google.com/patent/US7519795B2/en.

18. The foregoing Patents, namely the '014 Patent, the '526 Patent, the '478 Patent, the '472 Patent, and the '795 Patent are collectively referred to as "the Teleputers Patents."

19. The Teleputers Patents are valid, enforceable, and were duly issued in full compliance with Title 35 of the United States Code.

20. Defendant, at least by the date of this Original Complaint, is on notice of the Teleputers Patents.

ACCUSED INSTRUMENTALITIES

21. On information and belief, Defendants make, use, import, sell, and/or offer for sale a multitude of products and services as systems on chips ("SoC") that employ Arm Neon technology supporting the infringing instructions including, but not limited to: (1) the 88PA6270 SoC; (2) the 88PA6220 SoC; (3) the PXA1088 SoC; (4) the ARMADA 38x; and (5) the ThunderX3 (individually and collectively, the "Accused Instrumentalities"). On information and belief, the Accused Instrumentalities are made, used, sold, offered for sale, and/or imported in the United States by Defendants.

22. On information and belief, Defendants also make, use, import, sell, and/or offer for sale a multitude of products and services and provides, for example, the SPARC64 XII processor used in Fujitsu servers (including but not limited to SPARC M12-1, SPARC M12-2 and SPARC M12-2S) (the "'014 Accused Instrumentalities"). On information and belief, the '014 Accused Instrumentalities are made, used, sold, offered for sale, and/or imported in the United States by Defendants.

COUNT I
(Infringement of U.S. Patent No. 7,092,526B2)

23. Teleputers incorporates the above paragraphs by reference.

24. Defendant has been on notice of the '526 Patent at least as early as the date it received service of this Original Complaint.

25. On information and belief, Defendant has directly infringed and continue to infringe the '526 Patent by making, using, importing, selling, and/or, offering for sale the Accused Instrumentalities in the United States.

26. On information and belief, Defendant, with knowledge of the '526 Patent, indirectly infringes the '526 Patent by inducing others to infringe the '526 Patent. In particular, Defendant intends to induce customers to infringe the '526 Patent by encouraging customers to use the Accused Instrumentalities in a manner that results in infringement.

27. On information and belief, Defendant also induces others, including its customers, to infringe the '526 Patent by providing technical support for the use of the Accused Instrumentalities.

28. On information and belief, at all times Defendant owns and controls the operation of the Accused Instrumentalities in accordance with an end user license agreement.

29. On information and belief, the Accused Instrumentalities necessarily infringe one or more claims of the ‘478 Patent when used as intended.

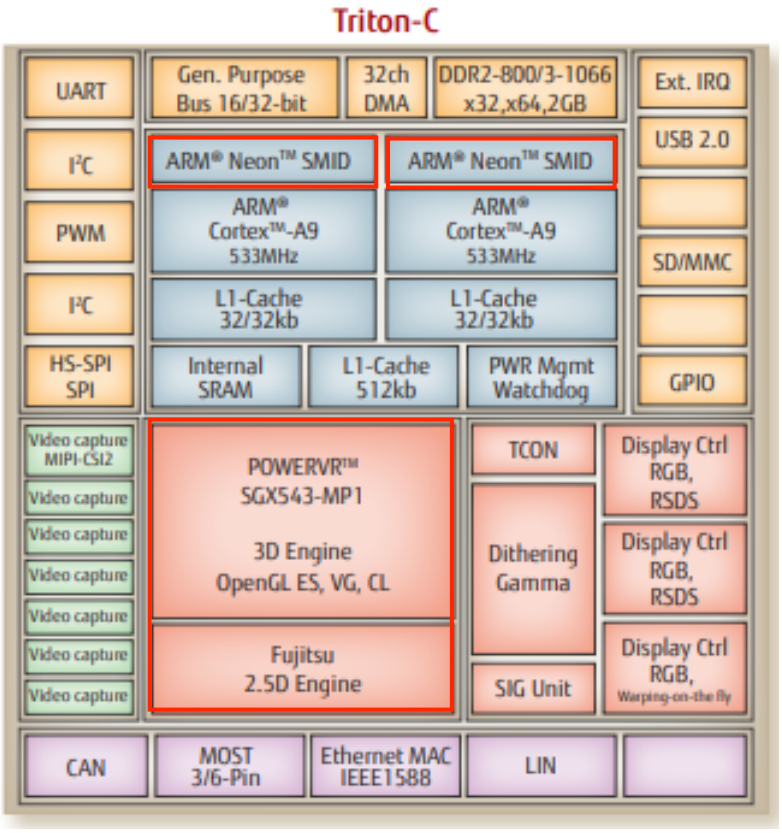
30. On information and belief, the Accused Instrumentalities infringe at least Claim 1 of the ’526 Patent by providing a method for permuting a two-dimensional (2D) data based on decomposing images and objects into atomic elements. For example, Defendant provides system-on-chip (including but not limited to 88PA6270 SoC, 88PA6220 SoC, PXA1088 SoC, ARMADA 38x, ARMADA 375, ARMADA LP and/or ThunderX3) solutions for parallel data processing. Defendant’s 88PA6270 SoC (used herein as an exemplary product) is used for class color and monochrome single or multi-function printers. The 88PA6270 SoC includes a quad core 1.2 GHz ARM A53 processor to handle all the application processing and Page Description Language (PDL) rendering requirements. Further, the 88PA6270 SoC (“programmable processor”) utilizes ARM Neon technology (a Single Instruction Multiple Data (SIMD) architecture) for improving video encoding and decoding, 2D/3D graphics and/or gaming experience. ARM Neon SIMD architecture provides permutation instructions to rearrange individual elements present in 2D/3D graphics. Further, upon information and belief, Defendant directly infringes the claim at least when it tests its SoCs. During such tests, Defendant utilizes the SoCs to perform permutation on the input data using permutation instructions available in ARM Neon SIMD ISA (Instruction Set Architecture).

MB86R24 "Triton" Series: The high-performance MB86R24 "Triton-C" combines the latest ARM Cortex-A9 dual CPU core with state-of-the-art, embedded 2.5D and 3D graphics cores. This third-generation application processor is the first device in Fujitsu's new "Blueline" family of high-performance GDCs.

The 3D core incorporates Imagination Technologies' POWERVR™ SGX543-MP1, which supports open standard API formats such as OpenGL ES 2.0. The POWERVR core uses Tile-Based Deferred Rendering (TBDR) for render processing, which reduces performance loads on the CPU and GPU, and increases system capacity. The high-end SoC also combines six video-capture inputs and three, independent, parallel display outputs.

The chip's architecture has been optimized for the simultaneous use of all functional blocks, virtually eliminating performance gaps. The device's harmonized structure permits the simultaneous rendering of independent 2.5D and 3D graphics, the capturing of multiple video streams, and the display of content to multiple sources.

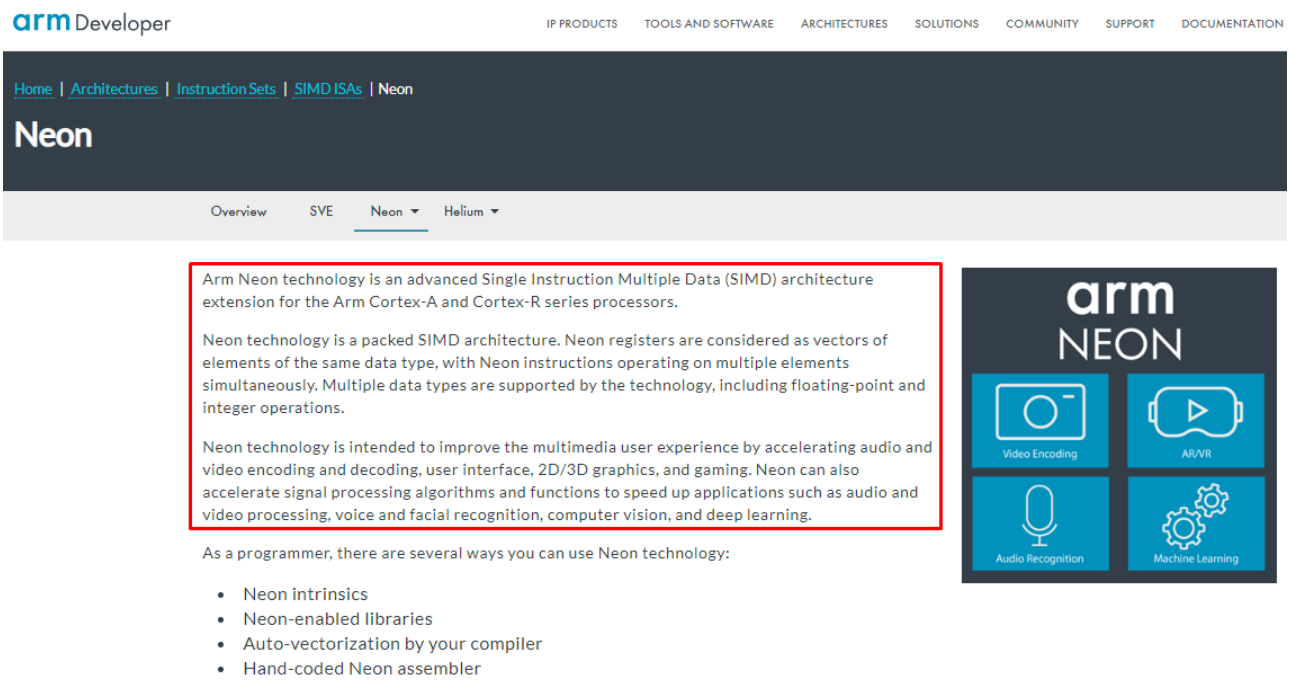
Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 3



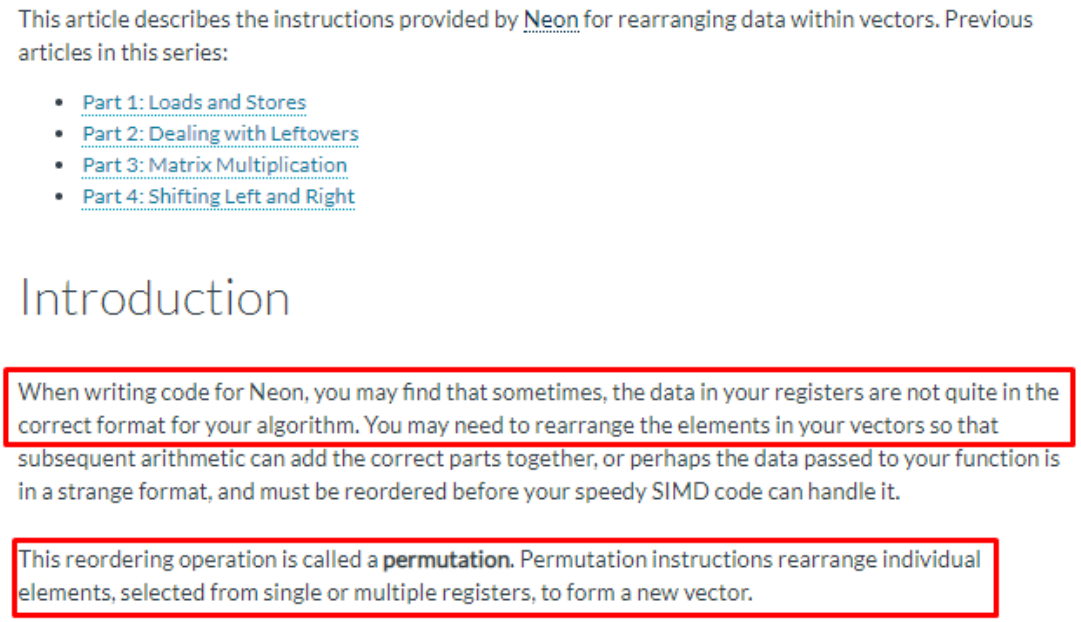
Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 3

Product	Description	Embedded Processor
MB88F33x "Indigo" Series	Sprite-based GDC with an APIX interface designed to be used in conjunction with "Jade D" or MB86R1x "Emerald" Series	No
MB86R03 "Jade –L"	2.5D/3D, DDR2, dual display/single capture. Interfaces: SD (1), I²C (2), I²S (3), PWM (2), UART (6), GPIO (24)	ARM926E
MB86R01 "Jade"	"Jade L" features plus USB, Media LB, IDE66	ARM926E
MB86R02 "Jade-D"	"Jade" plus an APIX (USB, IDE removed) dithering unit added to the display controller	ARM926E
MB86R11 "Emerald-L"	3D GDC core supporting OpenGL ES 2.0 plus new PixBlt engine for enhanced 2.5D processing. Four video-capture ports, with the ability to drive five displays. Interfaces include: Ethernet (1), SD (3), USB (2), I²C (5), I²S (4), PWM (12), UART (6), GPIO (25), CAN (2), SPI (2), QSPI (1)	ARM Cortex-A9
MB86R12 "Emerald-P"	Faster CPU (533 MHz) and graphics core (266MHz). Four high-speed APIX 2.0 ports – three outputs and one input. Rated for a –40 to +105° C operating range	ARM Cortex-A9
MB86R24 "Triton-C"	Dual-core ARM Cortex-A9 processors with Imagination's IMG543 3D graphics core combined with Fujitsu's high-performance 2.5D engine. Six video-capture inputs and three independent display controllers. Interfaces: SPI (3), Ethernet, USB 2.0, SDIO/MMC (1), UART (6), USART (5), I²S (2), I²C (4), and PWM (8)	ARM Cortex-A9

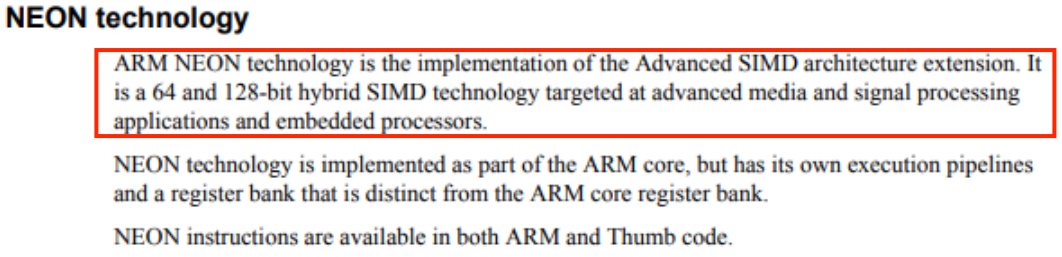
Source:https://www.fujitsu.com/us/Images/SPBG_GDC_Overview_PB.pdf, page 4



Source: https://developer.arm.com/architectures/instruction-sets/simd-isas/neon



Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors



NEON technology

ARM NEON technology is the implementation of the Advanced SIMD architecture extension. It is a 64 and 128-bit hybrid SIMD technology targeted at advanced media and signal processing applications and embedded processors.

NEON technology is implemented as part of the ARM core, but has its own execution pipelines and a register bank that is distinct from the ARM core register bank.

NEON instructions are available in both ARM and Thumb code.

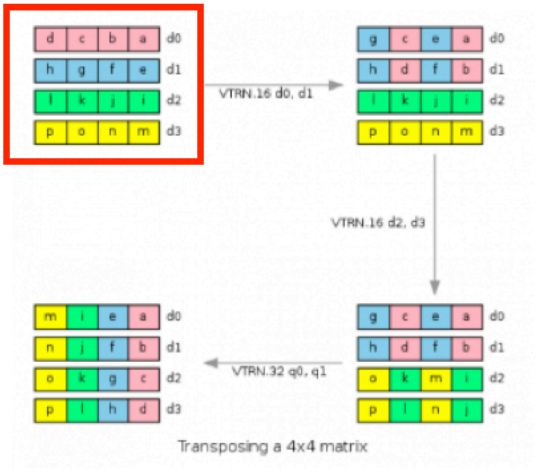
Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf,

page 40

31. Further, Defendant performs and induces others to perform the step of decomposing said two dimensional data into at least one atomic element said two dimensional data being located in at least one source register said at least one atomic element of said two dimensional data is a 2×2 matrix and said two dimensional data is decomposed into data elements in said matrix. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and decomposes two dimensional data (in the form of 4x4 matrix) into a 2x2 matrix. The 4x4 matrix consists of 16-bit elements ("atomic element"). The permutation instruction transposes 8, 16 or 32-bit elements between a pair of vectors. The permutation instruction is applied on the elements of the vectors by dividing it into 2x2 matrices. The two dimensional data is stored in at least one of the d0 and d1 vectors ("source registers").

Use multiple VTRN instructions to transpose larger matrices. For example, a 4x4 matrix consisting of 16-bit elements can be transposed using three VTRN instructions.



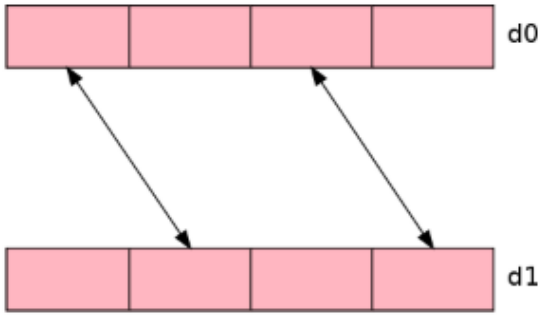
Transposing a 4x4 matrix

This is the same operation performed by VLD4 and VST4 after loading, or before storing, vectors. As they require fewer instructions, try to use these structured memory access features in preference to a sequence of VTRN instructions, where possible.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

14.131 VTRN

Vector Transpose.

Syntax

`VTRN{cond}.size Qd, Qm`

`VTRN{cond}.size Dd, Dm`

where:

cond
is an optional condition code.

size
must be one of 8, 16, or 32.

Qd, Qm
specifies the vectors, for a quadword operation.

Dd, Dm
specifies the vectors, for a doubleword operation.

Operation

VTRN treats the elements of its operand vectors as elements of 2 x 2 matrices, and transposes the matrices. The following figures show examples of the operation of VTRN:

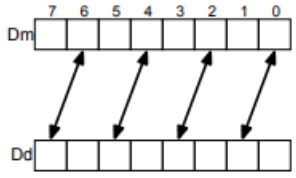


Figure 14-9 Operation of doubleword VTRN.8

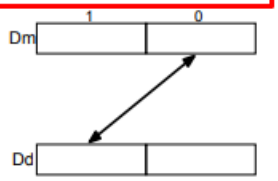


Figure 14-10 Operation of doubleword VTRN.32

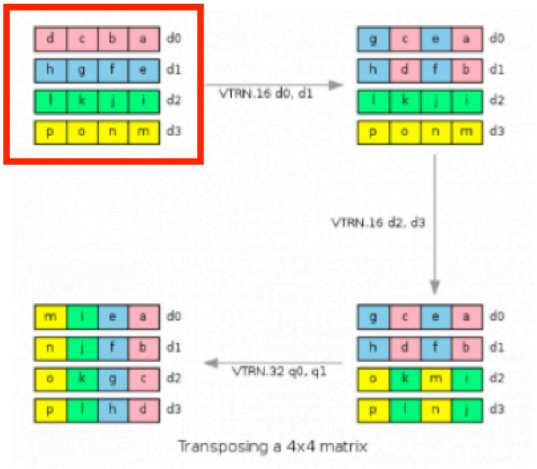
Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf,

page 734

32. Further, Defendant performs and induces others to perform the step of determining at least one permutation instruction for rearrangement of said data in said atomic element. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and decomposes two dimensional data (in the form of 4x4 matrix) into a 2x2 matrix. The permutation instruction transposes ("rearrangement") 8, 16 or 32-bit elements between a pair of vectors. The permutation instruction is applied on the elements of the vectors by dividing it into 2x2 matrices.

Use multiple VTRN instructions to transpose larger matrices. For example, a 4x4 matrix consisting of 16-bit elements can be transposed using three VTRN instructions.



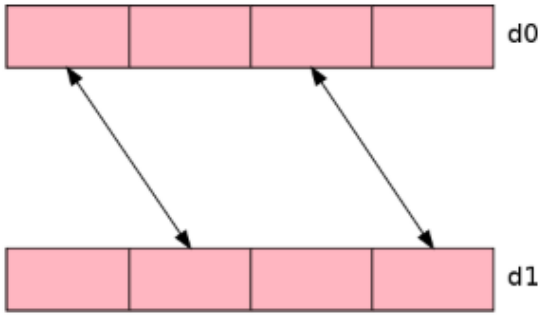
Transposing a 4x4 matrix

This is the same operation performed by VLD4 and VST4 after loading, or before storing, vectors. As they require fewer instructions, try to use these structured memory access features in preference to a sequence of VTRN instructions, where possible.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

14.131 VTRN

Vector Transpose.

Syntax

`VTRN{cond}.size Qd, Qm`

`VTRN{cond}.size Dd, Dm`

where:

cond
is an optional condition code.

size
must be one of 8, 16, or 32.

Qd, Qm
specifies the vectors, for a quadword operation.

Dd, Dm
specifies the vectors, for a doubleword operation.

Operation

VTRN treats the elements of its operand vectors as elements of 2 x 2 matrices, and transposes the matrices. The following figures show examples of the operation of VTRN:

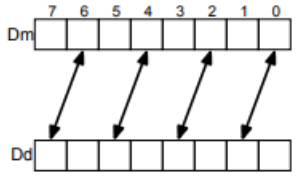


Figure 14-9 Operation of doubleword VTRN.8

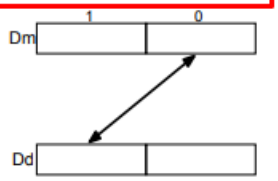


Figure 14-10 Operation of doubleword VTRN.32

Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf,

page 734

33. Further, Defendant performs and induces others to perform the step of said data elements being rearranged by said at least one permutation instruction, each of said data elements representing a subword having one or more bits. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and transposes ("rearrange") 8, 16 or 32-bit elements ("subwords") of the 2x2 matrix. The permutation instruction performs bit permutation on each element.

This article describes the instructions provided by Neon for rearranging data within vectors. Previous articles in this series:

- Part 1: Loads and Stores
- Part 2: Dealing with Leftovers
- Part 3: Matrix Multiplication
- Part 4: Shifting Left and Right

Introduction

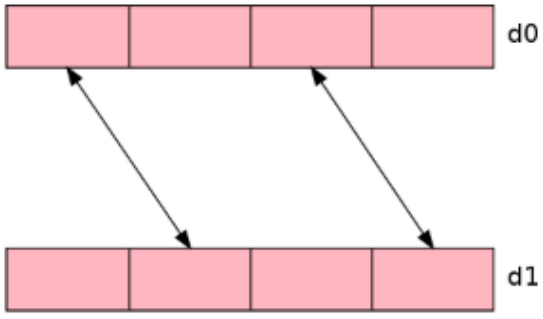
When writing code for Neon, you may find that sometimes, the data in your registers are not quite in the correct format for your algorithm. You may need to rearrange the elements in your vectors so that subsequent arithmetic can add the correct parts together, or perhaps the data passed to your function is in a strange format, and must be reordered before your speedy SIMD code can handle it.

This reordering operation is called a **permutation**. Permutation instructions rearrange individual elements, selected from single or multiple registers, to form a new vector.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

VTRN: Transpose

VTRN transposes 8, 16 or 32-bit elements between a pair of vectors. It treats the elements of the vectors as 2x2 matrices, and transposes each matrix.



VTRN.16 d0, d1

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

Operation

VTRN treats the elements of its operand vectors as elements of 2 x 2 matrices, and transposes the matrices.
The following figures show examples of the operation of VTRN:

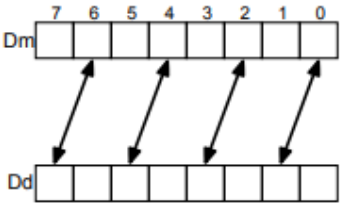


Figure 14-9 Operation of doubleword VTRN.8

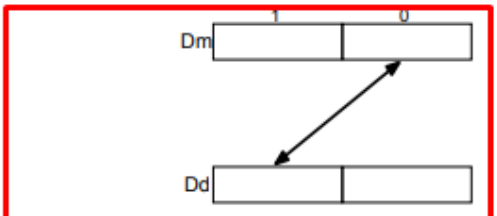


Figure 14-10 Operation of doubleword VTRN.32

Source:

http://infocenter.arm.com/help/topic/com.arm.doc.dui0473j/DUI0473J_armasm_user_guide.pdf, page 734

34. Further, Defendant performs and induces others to perform the step of applying said permutation instructions to said subwords and placing said permutated subwords into a destination register. For example, the MB86R24 Triton-C uses a permutation instruction (such as a VTRN instruction) and transposes 2x2 matrix elements to form a new vector ("placing said permutated subword into a destination register").

This article describes the instructions provided by Neon for rearranging data within vectors. Previous articles in this series:

- Part 1: Loads and Stores
- Part 2: Dealing with Leftovers
- Part 3: Matrix Multiplication
- Part 4: Shifting Left and Right

Introduction

When writing code for Neon, you may find that sometimes, the data in your registers are not quite in the correct format for your algorithm. You may need to rearrange the elements in your vectors so that subsequent arithmetic can add the correct parts together, or perhaps the data passed to your function is in a strange format, and must be reordered before your speedy SIMD code can handle it.

This reordering operation is called a **permutation**. Permutation instructions rearrange individual elements, selected from single or multiple registers, to form a new vector.

Source: https://community.arm.com/developer/ip-products/processors/b/processors-ip-blog/posts/coding-for-neon---part-5-rearranging-vectors

35. Defendant indirectly infringe the claim at least when Defendant's customers (such as device manufacturers which use Defendant's SoCs in their products) perform the method while testing their devices and when the devices are operated by end-users.

COUNT II
(Infringement of U.S. Patent No. 6,952,478B2)

36. Teleputers incorporates the above paragraphs by reference.

37. Defendant has been on notice of the '478 Patent at least as early as the date it received service of this Original Complaint.

38. On information and belief, Defendant has infringed and continue to infringe the '478 Patent by making, using, importing, selling, and/or, offering for sale the Accused Instrumentalities in the United States.

39. On information and belief, Defendant, with knowledge of the ’478 Patent, indirectly infringes the ’478 Patent by inducing others to infringe the ’478 Patent. In particular, Defendant intends to induce customers to infringe the ’478 Patent by encouraging customers to use the Accused Instrumentalities in a manner that results in infringement.

40. On information and belief, Defendant also induces others, including customers, to infringe the ’478 Patent by providing technical support for the use of the Accused Instrumentalities.

41. On information and belief, at all times Defendant owns and controls the operation of the Accused Instrumentalities in accordance with an end user license agreement.

42. On information and belief, the Accused Instrumentalities necessarily infringe one or more claims of the ’478 Patent when used as intended.

43. On information and belief, the Accused Instrumentalities infringe and induce others to infringe the ’478 Patent by providing a method for performing an arbitrary permutation of a source sequence of bits in a programmable processor. For example, Fujitsu provides a system-on-chip (including but not limited to MB86R1x, MB86R2x and MB86HD6x) solutions for parallel data processing. Defendant’s MB86R24 Triton-C SoC (used herein as an exemplary product) is used in Graphic Display Controllers. The MB86R24 Triton-C includes ARM Cortex-A9 dual CPU core, embedded 2.5D and 3D graphics cores. Further, the MB86R24 Triton-C SoC (“programmable processor”) utilizes Arm Neon technology (an advanced Single Instruction Multiple Data (SIMD) architecture) for improving audio/video encoding and decoding, 2.5D/3D graphics (“source sequence of bits”), and/or image/video processing. ARM Neon SIMD architecture provides permutation instructions to rearrange individual elements present in 2.5D/3D graphics.

44. For example, Defendant's 88PA6270 SoC (used herein as an exemplary product) is used for class color and monochrome single or multi-function printers. The 88PA6270 SoC includes a quad core 1.2 GHz ARM A53 processor to handle all the application processing and Page Description Language (PDL) rendering requirements.

45. Further, the 88PA6270 SoC ("programmable processor") utilizes ARM Neon technology (a Single Instruction Multiple Data (SIMD) architecture) for improving video encoding and decoding, 2D/3D graphics ("two dimensional (2-D) data"), and/or gaming experience. ARM Neon SIMD architecture provides permutation instructions to rearrange individual elements present in 2D/3D graphics.

46. Further, upon information and belief, Defendant directly infringes the claim at least when it tests its SoCs. During such tests, Defendant utilizes the SoCs to perform permutation on the input data using permutation instructions available in ARM Neon SIMD ISA (Instruction Set Architecture).

47. Further, Defendant indirectly infringes the claim at least when Defendant's customers (such as device manufacturers which use Defendant's SoCs in their products) perform the method while testing their devices and when the devices are operated by end-users.

48. Teleputers has been damaged by Defendant's infringement of the '478 Patent.

COUNT III
(Infringement of U.S. Patent No. 7,174,014)

49. Teleputers incorporates the above paragraphs by reference.

50. Defendant has been on notice of the '014 Patent at least as early as the date it received service of this Original Complaint.

51. On information and belief, Defendant has infringed and continue to infringe the ’014 Patent by making, using, importing, selling, and/or, offering for sale the ’014 Accused Instrumentalities in the United States.

52. On information and belief, Defendant, with knowledge of the ’014 Patent, indirectly infringes the ’014 Patent by inducing others to infringe the ’014 Patent. In particular, Defendant intends to induce customers to infringe the ’014 Patent by encouraging customers to use the ’014 Accused Instrumentalities in a manner that results in infringement.

53. On information and belief, Defendant also induces others, including customers, to infringe the ’014 Patent by providing technical support for the use of the ’014 Accused Instrumentalities.

54. On information and belief, at all times Defendant owns and controls the operation of the ’014 Accused Instrumentalities in accordance with an end user license agreement.

55. On information and belief, the Accused Instrumentalities necessarily infringe one or more claims of the ’478 Patent when used as intended.

56. On information and belief, the ’014 Accused Instrumentalities infringe and induce others to infringe the ’014 Patent by providing a method for performing an arbitrary permutation of a sequence of bits. For example, Defendant provides a SPARC64 XII processor used in Fujitsu servers (including but not limited to SPARC M12-1, SPARC M12-2 and SPARC M12-2S) for parallel data processing. Defendant’s SPARC M12-1 (used herein as an exemplary product) supports SPARC64 XII processor for mission-critical enterprise workloads and cloud computing. SPARC64 XII supports permutation instructions to permute the data in the register.

Fujitsu SPARC Servers Product Lineup

Models	Type	CPU	Max CPUs	Max Cores	Max Threads	Max Memory	Max Internal HDD
Fujitsu SPARC M12-1 	Rackmount (1U)	SPARC64™ XII 3.2GHz	1	6	48	1TB	7.2TB
Fujitsu SPARC M12-2 	Rackmount (4U)	SPARC64™ XII 3.9GHz	2	24	192	2TB	7.2TB
Fujitsu SPARC M12-2S 	Rackmount (4U)	SPARC64™ XII 4.25GHz	2	24	192	2TB	7.2TB
	Two dedicated racks		32	384	3072	32TB	115.2TB

Source: https://www.fujitsu.com/us/products/computing/servers/unix/sparc/lineup/

Source: https://www.fujitsu.com/global/products/computing/servers/unix/sparc/lineup/m12-1/

The Fujitsu SPARC M12-1 features:

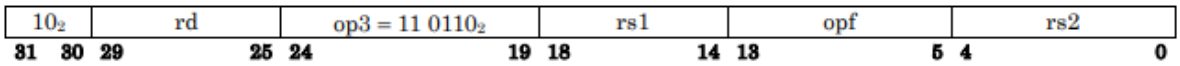
- The Fujitsu SPARC M12-1 server is designed to reduce total cost of ownership (TCO), rapidly deploy new business services, and reduce server sprawl by consolidating existing systems more cost-effectively and more reliably.
- CPU core-level Capacity on Demand allows granular and agile response to changes in business requirements. Fujitsu SPARC M12 Core Activation allows customers to start small and grow by supporting an initial minimum of two activated cores and step-by-step expansion in units of a single core.
- Dramatically improved Oracle Database processing and encryption performance with Fujitsu SPARC M12 Software on Chip functionality. By implementing dedicated per-core logic in each processor, Software on Chip offers vastly improved performance for tasks traditionally handled purely by software. Software on Chip functionalities include Single Instruction Multiple Data (SIMD) for in-memory processing, decimal floating-point operations, and encryption processing.
- Mainframe-class RAS (Reliability, Availablity, Serviceablity) features abound in Fujitsu SPARC M12 servers. Features to support the most business-critical processing include: automatic recovery with instruction retry, extended error-correcting code (ECC) protection, guaranteed data path integrity, configurable memory mirroring, redundant and hot-swappable system components and support for dual power feed implementations.
- Fujitsu SPARC M12 systems support the world's most advanced enterprise operating system: Oracle Solaris. Bare metal Solaris 10 and Solaris 11 are supported, as well as Solaris 8 and 9 in Oracle Solaris Legacy Containers, providing customers with exceptional investment protection and no-risk migration from previous server generations.
- To drive higher levels of system utilization and the resulting cost savings, Fujitsu SPARC M12 systems support no-cost Oracle VM Server for SPARC and Oracle Solaris Zones virtualization technologies, allowing for flexible server consolidation for the most demanding mixed-workload environments.

Fujitsu SPARC M12-1 is the ideal entry-level server for traditional enterprise-class workloads such as online transaction processing (OLTP), business intelligence and data warehousing (BIDW), enterprise resource planning (ERP), and customer relationship management (CRM), as well as new environments in cloud computing or big data processing.

Source: https://www.fujitsu.com/global/products/computing/servers/unix/sparc/lineup/m12-1/

7.148. Full Element Permutation

Opcode	opf	Operation	HPC-ACE Regs	HPC-ACE SIMD	Assembly Language Syntax
FEPERM32X[XII]	1 1000 0100$_2$	Sorts 32-bit data among double floating-point registers	✓	✓	`feperm32x` *freg$_{rs1}$*, *freg_or_fsimm*, *freg$_{rd}$*
FEPERM64X[XII]	1 1000 0101$_2$	Sorts 64-bit data among double floating-point registers	✓	✓	`feperm64x` *freg$_{rs1}$*, *freg_or_fsimm*, *freg$_{rd}$*



Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75

57. Further, Defendant performs and induces others to perform the step of inputting a source sequence of bits into a source register. For example, the SPARC M12-1 uses a permutation instruction (such as a FEPERM32X/FEPERM64X instruction) to permute the data bits. Fd[rs1] ("source register") contains the data bits to be permuted.

FEPERM32X and FEPERM64X are mainly used to permutate or mask the SIMD data. These instructions can be used in non-SIMD operations but the purpose is different from SIMD operations.

If xar_i = 0, FEPERM32X copies one of the 32-bit data ((1) - (3) as stated below) to Fd[rd]<63:32> according to Fd[rs2]<63, 32>, and to Fd[rd]<31:0> according to Fd[rs2]<31, 0>.

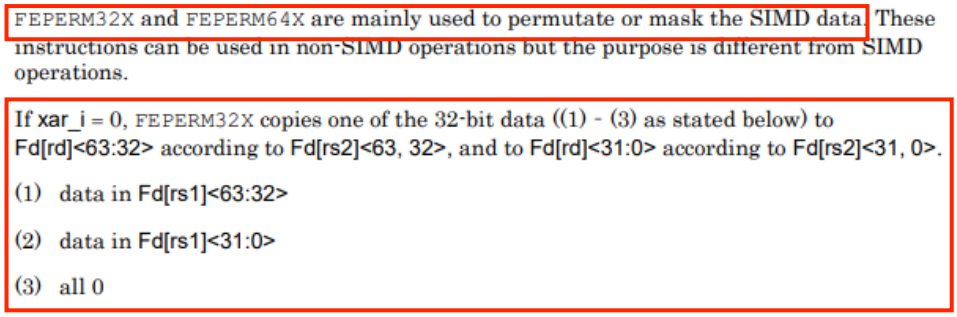
(1) data in Fd[rs1]<63:32>

(2) data in Fd[rs1]<31:0>

(3) all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf, page-75

58. Further, Defendant performs and induces others to perform the step of defining bit positions in said source sequence of bits to be permuted in said source register for a group of bits in a destination register. For example, the SPARC M12-1 implements a permutation instruction (such as a FEPERM32X/ FEPERM64X instruction) to permute the data bits. The data bits in the Fd[rs1] are permuted based on the data bits in Fd[rs2]. Therefore, Fd[rs2] determines the data bits in the Fd[rs1] that will be permuted and moved to the destination register Fd[rd]. If the 63rd bit of Fd[rs2] is 0 and the 32nd bit of the Fd[rs2] is 0, then the [63-32] bits of Fd[rs1] will be copied to [63-32] bit position in Fd[rd]. If the 63rd bit of Fd[rs2] is 0 and the 32nd bit of the Fd[rs2] is 1, then the [31-0] bits in Fd[rs1] will be copied to [63-32] bit position in Fd[rd]. Similarly, for the bit position [31-0] in Fd[rd], 31st and 0th bit of Fd[rs2] are checked.



FEPERM32X and FEPERM64X are mainly used to permutate or mask the SIMD data. These instructions can be used in non-SIMD operations but the purpose is different from SIMD operations.

If xar_i = 0, FEPERM32X copies one of the 32-bit data ((1) - (3) as stated below) to Fd[rd]<63:32> according to Fd[rs2]<63, 32>, and to Fd[rd]<31:0> according to Fd[rs2]<31, 0>.

(1) data in Fd[rs1]<63:32>

(2) data in Fd[rs1]<31:0>

(3) all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75

The behavior of FEPERM32X is described in Figure 7-8, Table 7-10, and Table 7-11. The value of Fd[rs2]<62:33, 30:1> is ignored.

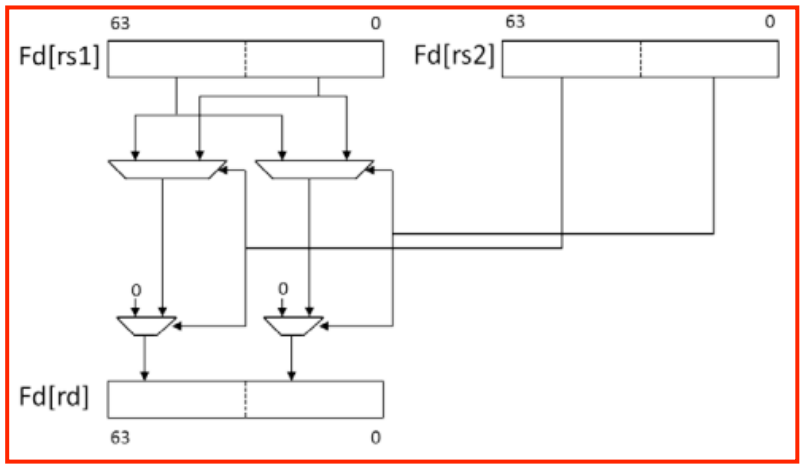


Figure 7-8 Behavior of FEPERM32X (xar_i = 0)

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75

Table 7-10 Results of FEPERM32X (Fd[rd]<63:32>, xar_i = 0)

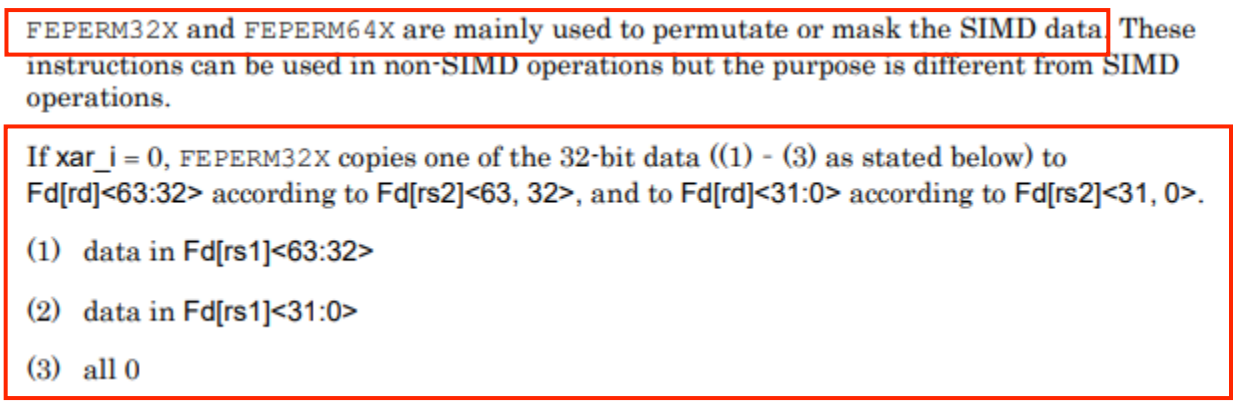
Fd[rs2]<63>	Fd[rs2]<32>	Fd[rd]<63:32>
0	0	Fd[rs1]<63:32>
	1	Fd[rs1]<31:0>
1	–	all 0

Table 7-11 Results of FEPERM32X (Fd[rd]<31:0>, xar_i = 0)

Fd[rs2]<31>	Fd[rs2]<0>	Fd[rd]<31:0>
0	0	Fd[rs1]<63:32>
	1	Fd[rs1]<31:0>
1	–	all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-76

59. Further, Defendant performs and induces others to perform the step of in response to a PPERM instruction inserting bits from said source sequence into said destination register as determined by said bit positions. For example, the SPARC M12-1 performs the FEPERM32X/FEPERM64X permutation instructions that are equivalent to the PPERM instruction as mentioned in the claimed invention. In response to the FEPERM32X/ FEPERM64X permutation instruction, the data bits in Fd[rs1] ("source sequence") are moved to the Fd[rd] ("destination register") according to the bit positions defined by Fd[rs2].



FEPERM32X and FEPERM64X are mainly used to permutate or mask the SIMD data. These instructions can be used in non-SIMD operations but the purpose is different from SIMD operations.

If xar_i = 0, FEPERM32X copies one of the 32-bit data ((1) - (3) as stated below) to Fd[rd]<63:32> according to Fd[rs2]<63, 32>, and to Fd[rd]<31:0> according to Fd[rs2]<31, 0>.

(1) data in Fd[rs1]<63:32>

(2) data in Fd[rs1]<31:0>

(3) all 0

Source:https://www.fujitsu.com/jp/documents/products/computing/servers/unix/sparc/downloads/documents/SPARC64XII-Specification.vol20.pdf page-75

60. Further, upon information and belief, Defendant directly infringes the claim at least when it tests its servers. During such tests, Defendant utilizes the SPARC XII processor to perform permutation on the input data using permutation instructions.

61. Further, Defendant indirectly infringes the claim at least when the server computers are operated by the customers. During such use, end users utilize the SPARC XII processor to perform permutation on the input data using permutation instructions.

62. Teleputers has been damaged by Defendant's infringement of the '014 Patent.

PRAYER FOR RELIEF

WHEREFORE, Teleputers respectfully requests the Court enter judgment against Defendants:

1. declaring that the Defendants have infringed each of the Patents-in-Suit;

2. awarding Teleputers its damages suffered as a result of Defendants' infringement of the Patents-in-Suit;

3. awarding Teleputers its costs, attorneys' fees, expenses, and interest;

4. awarding Teleputers ongoing post-trial royalties; and

5. granting Teleputers such further relief as the Court finds appropriate.

JURY DEMAND

Teleputers demands trial by jury, under Fed. R. Civ. P. 38.

Dated: July 15, 2020

Respectfully Submitted

/s/ Scott Fuller

M. Scott Fuller
Texas Bar No. 24036607
sfuller@ghiplaw.com
Thomas G. Fasone III
Texas Bar No. 00785382
tfasone@ghiplaw.com
GARTEISER HONEA, PLLC
119 W. Ferguson Street
Tyler, Texas 75702
Telephone: (903) 705-7420
Facsimile: (888) 908-4400

Raymond W. Mort, III
Texas State Bar No. 00791308
raymort@austinlaw.com
THE MORT LAW FIRM, PLLC
100 Congress Ave, Suite 2000
Austin, Texas 78701
Tel/Fax: (512) 865-7950

ATTORNEYS FOR PLAINTIFF TELEPUTERS LLC